Project Phase 2 Report – MyShell Remote

Submitted By: Bimarsha Adhikari (ba2422) and Sulav G. Shrestha (sgs9904)

Architecture and Design

- High-Level Structure
  - Reuse Phase 1 local shell core for parsing and execution: `parser.c/.h`, `executor.c/.h`, `error_handler.c/.h`, shared types in `myshell.h`.
  - Add a thin TCP client/server layer for remote execution: `myshell_server.c`, `myshell_client.c`, `network_utils.c/.h`.
  - Build with `Makefile` as a multi-binary project.
- Data and Control Flow
  1) Client connects to server over TCP (default host `127.0.0.1`, port `5050`).
  2) Client reads a line at a `$` prompt and sends it as a framed message.
  3) Server parses the line into a `pipeline_t` via `parse_command_line`, validates, then executes with `execute_pipeline`.
  4) Server captures combined stdout/stderr, frames the bytes, and sends them back.
  5) Client prints the response; session ends when user types `exit`.
- Framing/Protocol Choice
  - Length-prefixed messages: a 32-bit size followed by raw bytes. This avoids partial reads and simplifies buffering across process and network boundaries.
- Key Design Decisions
  - Preserve Phase 1 module boundaries; networking is an adapter, not a rewrite.
  - Errors printed by executed commands are captured and returned verbatim to ensure behavior parity with local shell, while server logs provide rubric-friendly markers.
  - Default `SIGPIPE` ignored in server, restored to default in children to mirror Unix pipeline semantics.
- File Organization
  - Shell core: `myshell.c` (local loop, retained), `parser.c/.h`, `executor.c/.h`, `error_handler.c/.h`, `myshell.h`.
  - Networking: `myshell_server.c`, `myshell_client.c`, `network_utils.c/.h`.
  - Build: `Makefile`.

Implementation Highlights

- Parsing and Structs

    - Tokenization with quotes/escapes: `tokenize` in `parser.c` produces `token_t` list with `quoted` metadata.
    - Pipeline builder: `parse_tokens` allocates `command_t[]` within `pipeline_t`, supporting `<`, `>`, `2>`, and `|`.
    - Memory lifecycle: `free_tokens`, `free_pipeline`.
- Execution Core
    - Single command: `execute_single_command` sets up redirections via `setup_redirections`, forks, and `execvp`s; parent `waitpid`s.
    - Pipelines: `execute_pipeline` creates `N-1` pipes with `create_pipes`, wires stdin/stdout using `dup2`, closes all FDs with `close_pipes`, and synchronizes via `wait_for_children`.
    - Error messages: child prints Bash-style "myshell: : command not found" for bare ENOENT.
- Server Runtime (`myshell_server.c`)
    - Per-connection loop `handle_client`:
        - Receives framed line with `recv_message`, trims via `trim_whitespace`, handles `exit`.
        - `run_shell_command` temporarily redirects server stdout/stderr into a pipe, executes the parsed pipeline, restores FDs, drains pipe to a buffer, and returns output plus status.
        - Logs steps with `[INFO]/[RECEIVED]/[EXECUTING]/[OUTPUT]/[ERROR]` markers; sends the exact output back using `send_message`.
    - Robust command-not-found detection for rubric logging using `extract_missing_command` and `is_actual_command_name` without altering client-visible bytes.
- Client Runtime (`myshell_client.c`)
    - Connects, reads user input, sends framed messages, prints server replies unchanged.

Execution Instructions

make clean && make

Terminal A – start server (default 5050)

./myshell_server 5050

Terminal B – start client and connect to localhost

./myshell_client 127.0.0.1 5050

Examples on the client

$ ls -l

$ echo hello | tr a-z A-Z

$ cat < input.txt | grep foo | sort | uniq > out.txt

$ exit

Clean build artifacts

make clean

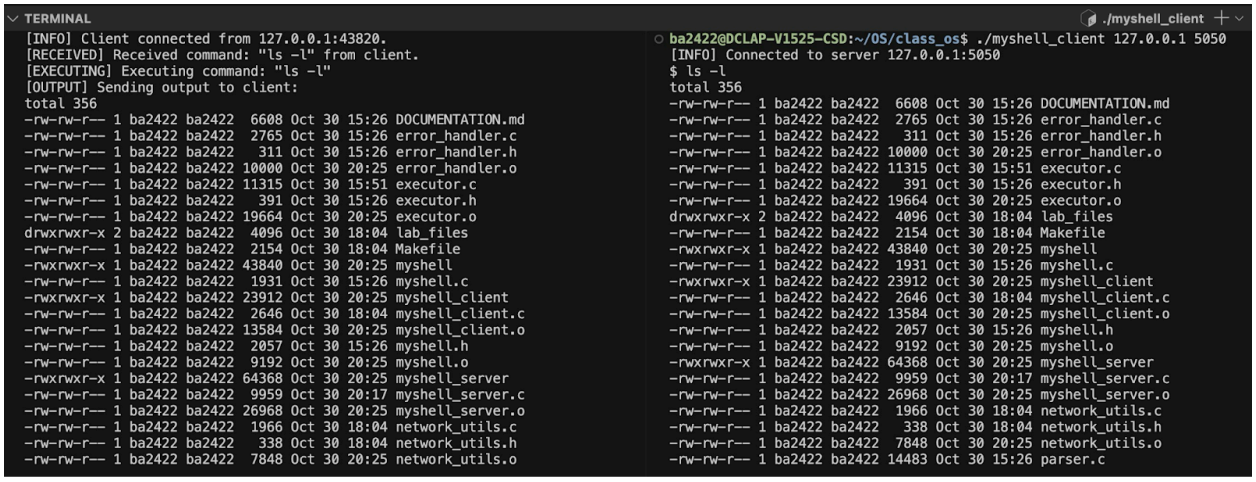
Challenges

- Pipe EOF/Hang Risks
  - Resolved by closing all unused pipe FDs in every child and in the parent (`close_pipes`), ensuring readers observe EOF.
- Quote/Escape Handling
  - Implemented stateful tokenizer for single/double quotes and common escapes; emits unmatched-quote parse errors with `print_parse_error`.
- Network Framing and Mixed Output
  - Used size-prefixed frames to carry arbitrary bytes from combined stdout/stderr; client prints exactly what the local shell would display.

Division of Tasks

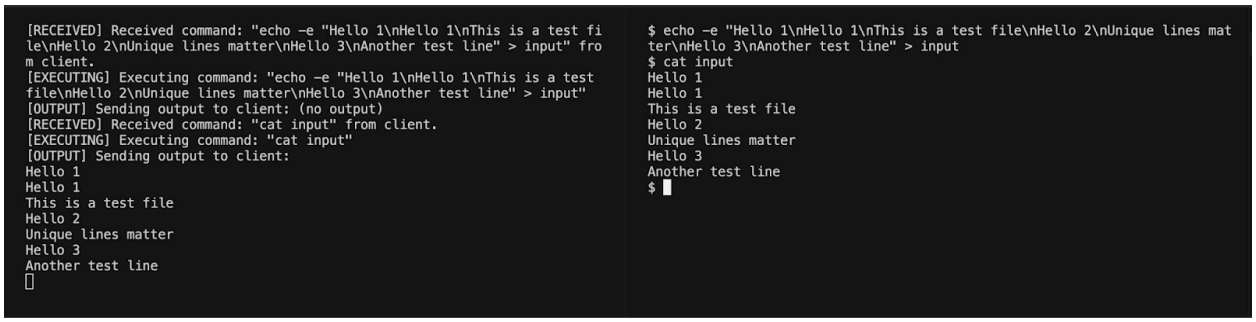
- Networking, server/client integration, framing helpers: Bimarsha Adhikari (ba2422)
- Shell parsing/execution modules and validation: Sulav G Shrestha (sgs9904)
- Build system and documentation: Both

Testing (Server | Client view)



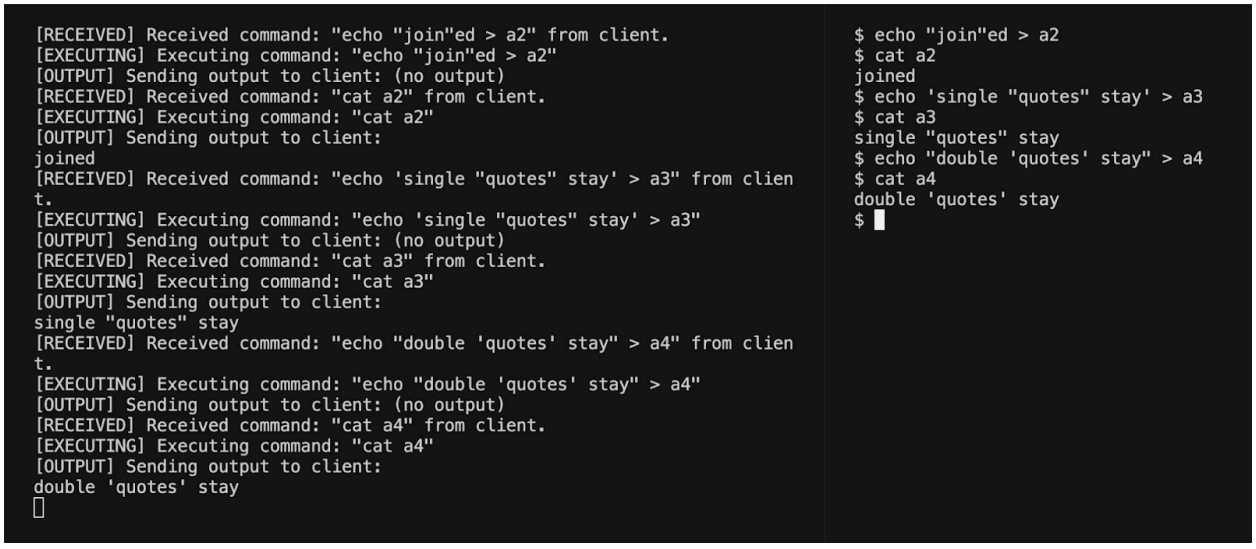
```
TERMINAL                                                                    ./myshell_client
[INFO] Client connected from 127.0.0.1:43820.
[RECEIVED] Received command: "ls -l" from client.
[EXECUTING] Executing command: "ls -l"
[OUTPUT] Sending output to client:
total 356
-rw-rw-r-- 1 ba2422 ba2422  6608 Oct 30 15:26 DOCUMENTATION.md
-rw-rw-r-- 1 ba2422 ba2422  2765 Oct 30 15:26 error_handler.c
-rw-rw-r-- 1 ba2422 ba2422   311 Oct 30 15:26 error_handler.h
-rw-rw-r-- 1 ba2422 ba2422 10000 Oct 30 20:25 error_handler.o
-rw-rw-r-- 1 ba2422 ba2422 11315 Oct 30 15:51 executor.c
-rw-rw-r-- 1 ba2422 ba2422   391 Oct 30 15:26 executor.h
-rw-rw-r-- 1 ba2422 ba2422 19664 Oct 30 20:25 executor.o
drwxrwxr-x 2 ba2422 ba2422  4096 Oct 30 18:04 lab_files
-rw-rw-r-- 1 ba2422 ba2422  2154 Oct 30 18:04 Makefile
-rwxrwxr-x 1 ba2422 ba2422 43840 Oct 30 20:25 myshell
-rw-rw-r-- 1 ba2422 ba2422  1931 Oct 30 15:26 myshell.c
-rwxrwxr-x 1 ba2422 ba2422 23912 Oct 30 20:25 myshell_client
-rw-rw-r-- 1 ba2422 ba2422  2646 Oct 30 18:04 myshell_client.c
-rw-rw-r-- 1 ba2422 ba2422 13584 Oct 30 20:25 myshell_client.o
-rw-rw-r-- 1 ba2422 ba2422  2057 Oct 30 15:26 myshell.h
-rw-rw-r-- 1 ba2422 ba2422  9192 Oct 30 20:25 myshell.o
-rwxrwxr-x 1 ba2422 ba2422 64368 Oct 30 20:25 myshell_server
-rw-rw-r-- 1 ba2422 ba2422  9959 Oct 30 20:17 myshell_server.c
-rw-rw-r-- 1 ba2422 ba2422 26968 Oct 30 20:25 myshell_server.o
-rw-rw-r-- 1 ba2422 ba2422  1966 Oct 30 18:04 network_utils.c
-rw-rw-r-- 1 ba2422 ba2422   338 Oct 30 18:04 network_utils.h
-rw-rw-r-- 1 ba2422 ba2422  7848 Oct 30 20:25 network_utils.o
```

```
ba2422@DCLAP-V1525-CSD:~/OS/class_os$ ./myshell_client 127.0.0.1 5050
[INFO] Connected to server 127.0.0.1:5050
$ ls -l
total 356
-rw-rw-r-- 1 ba2422 ba2422  6608 Oct 30 15:26 DOCUMENTATION.md
-rw-rw-r-- 1 ba2422 ba2422  2765 Oct 30 15:26 error_handler.c
-rw-rw-r-- 1 ba2422 ba2422   311 Oct 30 15:26 error_handler.h
-rw-rw-r-- 1 ba2422 ba2422 10000 Oct 30 20:25 error_handler.o
-rw-rw-r-- 1 ba2422 ba2422 11315 Oct 30 15:51 executor.c
-rw-rw-r-- 1 ba2422 ba2422   391 Oct 30 15:26 executor.h
-rw-rw-r-- 1 ba2422 ba2422 19664 Oct 30 20:25 executor.o
drwxrwxr-x 2 ba2422 ba2422  4096 Oct 30 18:04 lab_files
-rw-rw-r-- 1 ba2422 ba2422  2154 Oct 30 18:04 Makefile
-rwxrwxr-x 1 ba2422 ba2422 43840 Oct 30 20:25 myshell
-rw-rw-r-- 1 ba2422 ba2422  1931 Oct 30 15:26 myshell.c
-rwxrwxr-x 1 ba2422 ba2422 23912 Oct 30 20:25 myshell_client
-rw-rw-r-- 1 ba2422 ba2422  2646 Oct 30 18:04 myshell_client.c
-rw-rw-r-- 1 ba2422 ba2422 13584 Oct 30 20:25 myshell_client.o
-rw-rw-r-- 1 ba2422 ba2422  2057 Oct 30 15:26 myshell.h
-rw-rw-r-- 1 ba2422 ba2422  9192 Oct 30 20:25 myshell.o
-rwxrwxr-x 1 ba2422 ba2422 64368 Oct 30 20:25 myshell_server
-rw-rw-r-- 1 ba2422 ba2422  9959 Oct 30 20:17 myshell_server.c
-rw-rw-r-- 1 ba2422 ba2422 26968 Oct 30 20:25 myshell_server.o
-rw-rw-r-- 1 ba2422 ba2422  1966 Oct 30 18:04 network_utils.c
-rw-rw-r-- 1 ba2422 ba2422   338 Oct 30 18:04 network_utils.h
-rw-rw-r-- 1 ba2422 ba2422  7848 Oct 30 20:25 network_utils.o
-rw-rw-r-- 1 ba2422 ba2422 14483 Oct 30 15:26 parser.c
```



```
[RECEIVED] Received command: "echo -e "Hello 1\nHello 1\nThis is a test fi
le\nHello 2\nUnique lines matter\nHello 3\nAnother test line" > input" fro
m client.
[EXECUTING] Executing command: "echo -e "Hello 1\nHello 1\nThis is a test
file\nHello 2\nUnique lines matter\nHello 3\nAnother test line" > input"
[OUTPUT] Sending output to client: (no output)
[RECEIVED] Received command: "cat input" from client.
[EXECUTING] Executing command: "cat input"
[OUTPUT] Sending output to client:
Hello 1
Hello 1
This is a test file
Hello 2
Unique lines matter
Hello 3
Another test line
```

```
$ echo -e "Hello 1\nHello 1\nThis is a test file\nHello 2\nUnique lines mat
ter\nHello 3\nAnother test line" > input
$ cat input
Hello 1
Hello 1
This is a test file
Hello 2
Unique lines matter
Hello 3
Another test line
$
```



```
[RECEIVED] Received command: "echo "join"ed > a2" from client.
[EXECUTING] Executing command: "echo "join"ed > a2"
[OUTPUT] Sending output to client: (no output)
[RECEIVED] Received command: "cat a2" from client.
[EXECUTING] Executing command: "cat a2"
[OUTPUT] Sending output to client:
joined
[RECEIVED] Received command: "echo 'single "quotes" stay' > a3" from clien
t.
[EXECUTING] Executing command: "echo 'single "quotes" stay' > a3"
[OUTPUT] Sending output to client: (no output)
[RECEIVED] Received command: "cat a3" from client.
[EXECUTING] Executing command: "cat a3"
[OUTPUT] Sending output to client:
single "quotes" stay
[RECEIVED] Received command: "echo "double 'quotes' stay" > a4" from clien
t.
[EXECUTING] Executing command: "echo "double 'quotes' stay" > a4"
[OUTPUT] Sending output to client: (no output)
[RECEIVED] Received command: "cat a4" from client.
[EXECUTING] Executing command: "cat a4"
[OUTPUT] Sending output to client:
double 'quotes' stay
```

```
$ echo "join"ed > a2
$ cat a2
joined
$ echo 'single "quotes" stay' > a3
$ cat a3
single "quotes" stay
$ echo "double 'quotes' stay" > a4
$ cat a4
double 'quotes' stay
$
```

```
[RECEIVED] Received command: "echo *.o > a6" from client.
[EXECUTING] Executing command: "echo *.o > a6"
[OUTPUT] Sending output to client: (no output)
[RECEIVED] Received command: "cat a6" from client.
[EXECUTING] Executing command: "cat a6"
[OUTPUT] Sending output to client:
error_handler.o executor.o myshell.o myshell_client.o myshell_server.o net
work_utils.o parser.o
```

```
$ echo *.o > a6
$ cat a6
error_handler.o executor.o myshell.o myshell_client.o myshell_server.o netw
ork_utils.o parser.o
$
```

```
$ unknowncmd
myshell: unknowncmd: command not found
$ ls invalid_folder | wc -l 2> error
ls: cannot access 'invalid_folder': No such file or directory
0
$ ls non_existent_folder | grep "somefile" | invalid_command 2> error
ls: cannot access 'non_existent_folder': No such file or directory
$ ls | invalid_command | wc -l
myshell: invalid_command: command not found
0
$ grep "hello" < non_existent.txt
myshell: non_existent.txt: No such file or directory
$ cat <
myshell: Input file not specified.
$ ls >
myshell: Output file not specified.
$ ls non_existent 2>
myshell: Error output file not specified.
$ ls |
myshell: Command missing after pipe.
$ ls || wc -l
myshell: Empty command between pipes.
$ cat < input | sort >
myshell: Output file not specified.
$
```